Программирование

Python

Лекция 7

Кафедра ИВТ и ПМ
ЗабГУ

2017

План

Outline

Прошлые темы

- Что будет выведено на экран?

```
l1 = [1,2,3,4]
l2 = l1 * 2
print(l2)
```

Прошлые темы

- Что будет выведено на экран?

```
l1 = [1,2,3,4]
l2 = l1 * 2
print(l2)
```

```
[1, 2, 3, 1, 2, 3]
```

- Как удалить элементы с индексами от 1 до 3 из списка?

```
del l2[1:4]
```

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

Прошлые темы

- Что будет выведено на экран?

```
l1 = [1,2,3,4]
l2 = l1 * 2
print(l2)
```

```
[1, 2, 3, 1, 2, 3]
```

- Как удалить элементы с индексами от 1 до 3 из списка?

```
del l2[1:4]
```

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

```
[10, 20, 999]
```

Прошлые темы

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

Прошлые темы

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

```
[10, 20, 999]
```

- Как создать *копию* списка?

Прошлые темы

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

```
[10, 20, 999]
```

- Как создать *копию* списка?

```
l2 = l1.copy()
```

или

```
l3 = l1[:]
```

Прошлые темы

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

```
[10, 20, 999]
```

- Как создать *копию* списка?

```
l2 = l1.copy()
```

или

```
l3 = l1[:]
```

- Что означает **[:]** в предыдущем примере?

Прошлые темы

- Что будет выведено на экран?

```
l1 = [10, 20, 30]
l2 = l1
l1[2] = 999
print(l2)
```

```
[10, 20, 999]
```

- Как создать *копию* списка?

```
l2 = l1.copy()
```

или

```
l3 = l1[:]
```

- Что означает **[:]** в предыдущем примере?
  Это срез содержащий элементы списка от начала до конца.

Прошлые темы

Как определить находится ли слово в строке?

S = "Один Бритый Англичанин Финики Жевал, Как Морковь"

Прошлые темы

Как определить находится ли слово в строке?

S = "Один Бритый Англичанин Финики Жевал, Как Морковь"

```
if "Финики" in S:
    print("Есть такое слово!")
else:
    print("Нет такого слова")
```

Прошлые темы

Как добавить Юникод-символ с кодом ?

S = "Один Бритый Англичанин Финики Жевал, Как Морковь"

Прошлые темы

Как добавить Юникод-символ с кодом ?

S = "Один Бритый Англичанин Финики Жевал, Как Морковь"

```
S = "blah blah blah \u2211"
```

или

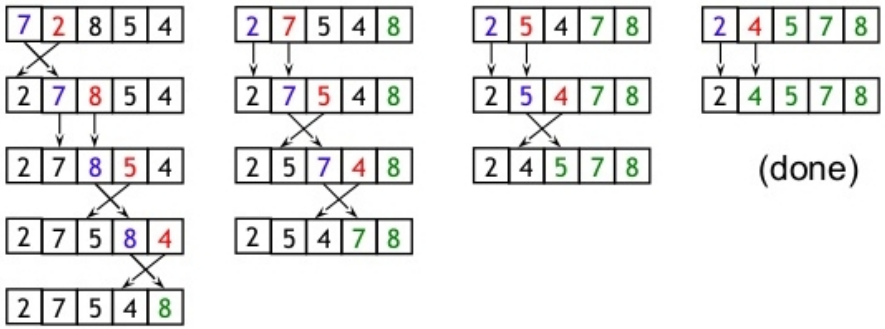
```
S = "blah blah blah " + chr(0x2211)
```

Прошлые темы

- Как работает сортировка выбором?

Прошлые темы

- Как работает сортировка выбором?



Прошлые темы

- Как работает алгоритм пузырьковой сортировки?

Прошлые темы

- Как работает алгоритм пузырьковой сортировки?

Прошлые темы

- `print( list( range(5) ) )`

Прошлые темы

- `print( list( range(5) ) )`

  `[0, 1, 2, 3, 4]`
- `print( list( range(1,6) ) )`

Прошлые темы

- `print( list( range(5) ) )`

  `[0, 1, 2, 3, 4]`
- `print( list( range(1,6) ) )`

  `[1, 2, 3, 4, 5]`
- `print( list( range(5,17, 3) ) )`

Прошлые темы

- `print( list( range(5) ) )`

  `[0, 1, 2, 3, 4]`
- `print( list( range(1,6) ) )`

  `[1, 2, 3, 4, 5]`
- `print( list( range(5,17, 3) ) )`

  `[5, 8, 11, 14]`

Прошлые темы

- Какой из этих двух сниплетов лучше? Почему?

```
L = [0]*N

for i in range(1,N*2+1):
    L[i] = (2*i)**2 + L[i]
```

```
L = []
sum = 0
for i in range(2,N*2+1,2):
    sum += L[-1]
    L += (2*i)**2
```

Прошлые темы

- Какой из этих двух сниплетов лучше? Почему?

```
L = [0]*N

for i in range(1,N*2+1):
    L[i] = (2*i)**2 + L[i]
```

```
L = []
sum = 0
for i in range(2,N*2+1,2):
    sum += L[-1]
    L += (2*i)**2
```

Правый вариант лучше:
операция индексирования списка - медленная;
здесь нет необходимости создавать список заренее.

Прошлые темы

- Какой список будет создать в результате?

```
L = [ i % 2 for i in range(10) ]
```

Прошлые темы

- Какой список будет создать в результате?

```
L = [ i % 2 for i in range(10) ]
```

```
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

- Какой список будет создан в результате?

```
L = [ i*j for i,j in [(1,2),(3,4)] ]
```

Прошлые темы

- Какой список будет создать в результате?

```
L = [ i % 2 for i in range(10) ]
```

```
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

- Какой список будет создан в результате?

```
L = [ i*j for i,j in [(1,2),(3,4)] ]
```

```
[2, 12]
```

Прошлые темы

- Как создать список из случайных целых чисел?

  ```
  L = [randint(0, 100) for i in range(10)]
  ```

- Как создать список из случайных чётных чисел?

Прошлые темы

- Как создать список из случайных целых чисел?

```
L = [randint(0, 100) for i in range(10)]
```

- Как создать список из случайных чётных чисел?

```
L = [randint(0, 100)*2 for i in range(10)]
```

Прошлые темы

- Как найти сумму каждого второго элемента в списке?

Прошлые темы

- Как найти сумму каждого второго элемента в списке?

```
L = [1,2,3,4,8]
sum = 0
for e in L[::2]:
    sum += e
```

sum = 12

Outline

Множества

- Что такое множество (в математике)?

Множества

- Что такое множество (в математике)?
- Может ли множество содержать один и тот же элемент дважды?

Множества

- Что такое множество (в математике)?
- Может ли множество содержать один и тот же элемент дважды?
- Определено ли отношение порядка для элементов множества?

Множества

- Что такое множество (в математике)?
- Может ли множество содержать один и тот же элемент дважды?
- Определено ли отношение порядка для элементов множества?

Множества

Создание множества

- Пустое множество

  ```
  s0 = set()
  ```

- Непустое множество

  ```
  s0 = {1, 3, -8, 4}
  ```

Используя пустые фигурные скобки создать пустое множество нельзя.

Множества

Создание множества из другого набора данных

- создание множества из списка

```
s1 = set( [1, 0, -8, 22] )
```

- создание множества из кортежа

```
s2 = set( (1, 0, -8, 22) )
```

- создание множества из строки

```
s3 = set('quick fox')

# s3 = {' ', 'c', 'f', 'i', 'k', 'o', 'q', 'u', 'x'}
```

Каждый символ строки будет считаться отдельным элементом множества.

Множества

Множество может содержать элементы любого типа

```
s4 = set( [1, 7.125, "Hello"] )
```

Множество не допускат хранение одинаковых элементов

```
s4 = set( ["A", 1, True, False, "A"] )

# s4 -> {False, 1, 'A'}
```

Добавление элементов в множество происходит в указанном порядке.
True не добавится потому, что 1 уже есть в множестве.

Множества

Операции с множествами

- получение размера множества

  ```
  len( s0 )
  ```

- Проверка вхождения элемента в множество

  ```
  0 in {-6,2,0, 8}  # True
  ```

- Сравнение множеств

  ```
  set1 == set2
  ```

  True если все элементы множества set1 совпадают с элементами в set2

Множества

Операции с множествами

- Является ли множество set подмножеством other?

  `set.issubset(other)`
- Является лм множество set надмножеством other?

  `set.issuperset(other)`
- Все ли элементов множеств различны?

  `set.isdisjoint(other)`

  истина, если set и other не имеют общих элементов.

Множества

Операции с множествами

- Пересечение множеств

  `newset = set.intersection(other, ...)`

  аналогично

  `newset = set & other & ...`

- Является лм множество set надмножеством other?

  `newset = set.union(other, ...)`

  аналогично minttext|newset = set | other | ...|

- Симметрическая разность

  `newset = set.symmetric_difference(other)`

  аналогично

  `newset = set ^ other`

Множества

Операции с множествами

- set.add(elem) - добавляет элемент в множество.
- set.remove(elem) - удаляет элемент из множества. KeyError, если такого элемента не существует.
- set.discard(elem) - удаляет элемент, если он находится в множестве.
- set.pop() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
- set.clear() - очистка множества.

Множества

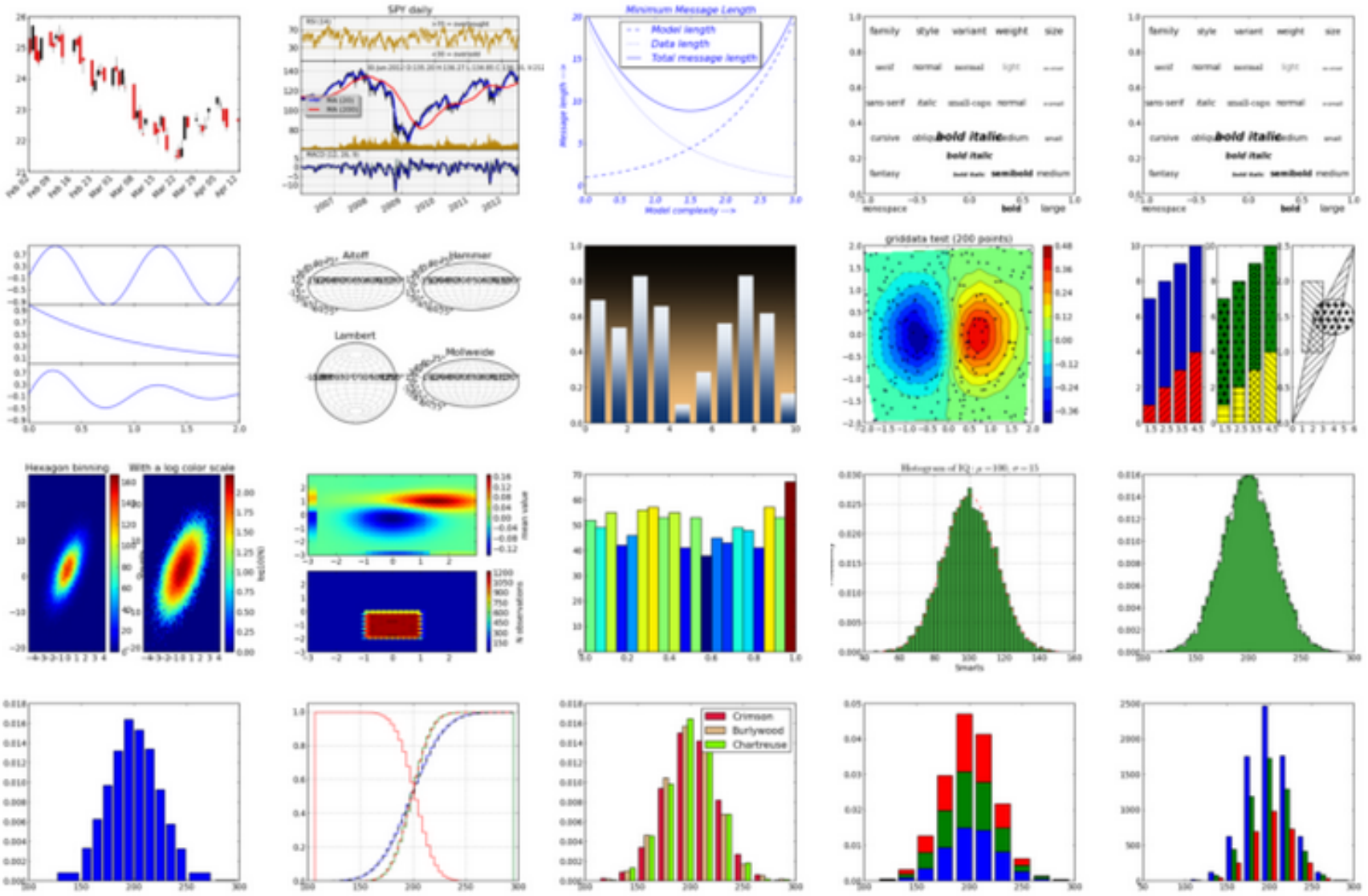
Когда использовать множества?
Когда нужна коллекция элементов без повторений, в которой порядок не имеет значения.

Быстродействие

- операция проверки вхождения элемента в множество быстрее такой же операции для списка или кортежа.
- Но операция перебора множества медленнее чем аналогичная для списка.

Outline

Пакет matplotlib



Пакет matplotlib

Пакет **matplotlib** содержит модули предназначенные для построения диаграмм и графиков.

Модуль pyplot из этого пакета предназначен непосредственно для построения графиков. Этого модуля будет достаточно для построения относительно простых графиков.

Остальные модули пакета matplotlib содержат преимущественно функции для гибкой настройки вида графиков, осей, подписей к осям, компоновки нескольких графиков на одном листе и т.п.

Пакет matplotlib

```
from matplotlib.pyplot import plot
```

Основные функции модуля pyplot
Создание графика **plot** - функция с переменным числом параметров. Если некоторые параметры не указаны, то им задаётся значение по-умолчанию.

- **plot(y-list)** - создаёт график в памяти программы
  y-list - список ординат (значений y).
  В качестве абсцисс (значений y) используются номера значений из списка y-list, т.е. индексы.

```
help(plot)
```

Пакет matplotlib

- **plot(x-list, y-list)**

  x-list - список абсцисс (значений x).
  y-list - список ординат (знаений y).
  Длинны списков x-list и y-list должны быть одинаковы.

Пакет matplotlib

Дополнительные параметры

- **plot(x-list, y-list, style)**
  style - стиль графика. Определяет цвет, вид кривой и точек
  - \- прямая линия
  - \- - пунктирная линия
  - **.** только точки
  - **v** треугольники вместо точек

  Цвета
  - **'b'** blue
  - **'g'** green
  - **'r'** red
  - **'k'** black

Дополнительную информацию о стилях см. в документации

Пакет matplotlib

Дополнительные параметры

- **plot(x-list, y-list, style, label)**
  label - подпись к графику. По-умолчанию не показывается.
- **plot(x-list, y-list, style, label, linewidth)**
  linewidth - толщина линии. По-умолчанию - 1.

Пакет matplotlib

```
from matplotlib.pyplot import grid, xlabel, ylabel, legend
```

- **grid(True)** - "включает"координатную сетку.
  Шаг сетки выбирается автоматически.
- **xlabel**("подпись") - добавляет название для оси x
- **ylabel**("подпись") - добавляет название для оси y
- **legend(loc)** - добавляет к графикам пояснения (легенду)
  loc - положение блока с пояснениями. По-умолчанию - справа сверху.
  Для автоматического выбора положения следует задать параметр loc = 'best'

Пакет matplotlib

```
from matplotlib.pyplot import show, savefig
```

- **show()** - создаёт и показывает окно содержащие построенные функциями plot() графики.
  Оси координат строятся строятся автоматически, масштаб выбирается в зависимости от ширины и высоты графика.
- **savefig(filename [, dpi])** - сохраняет изображение графика в файл. Графический формат определяется по расширению файла.
  filename - имя файла
  dpi - количество точек на дюйм (DPI)

Пакет matplotlib

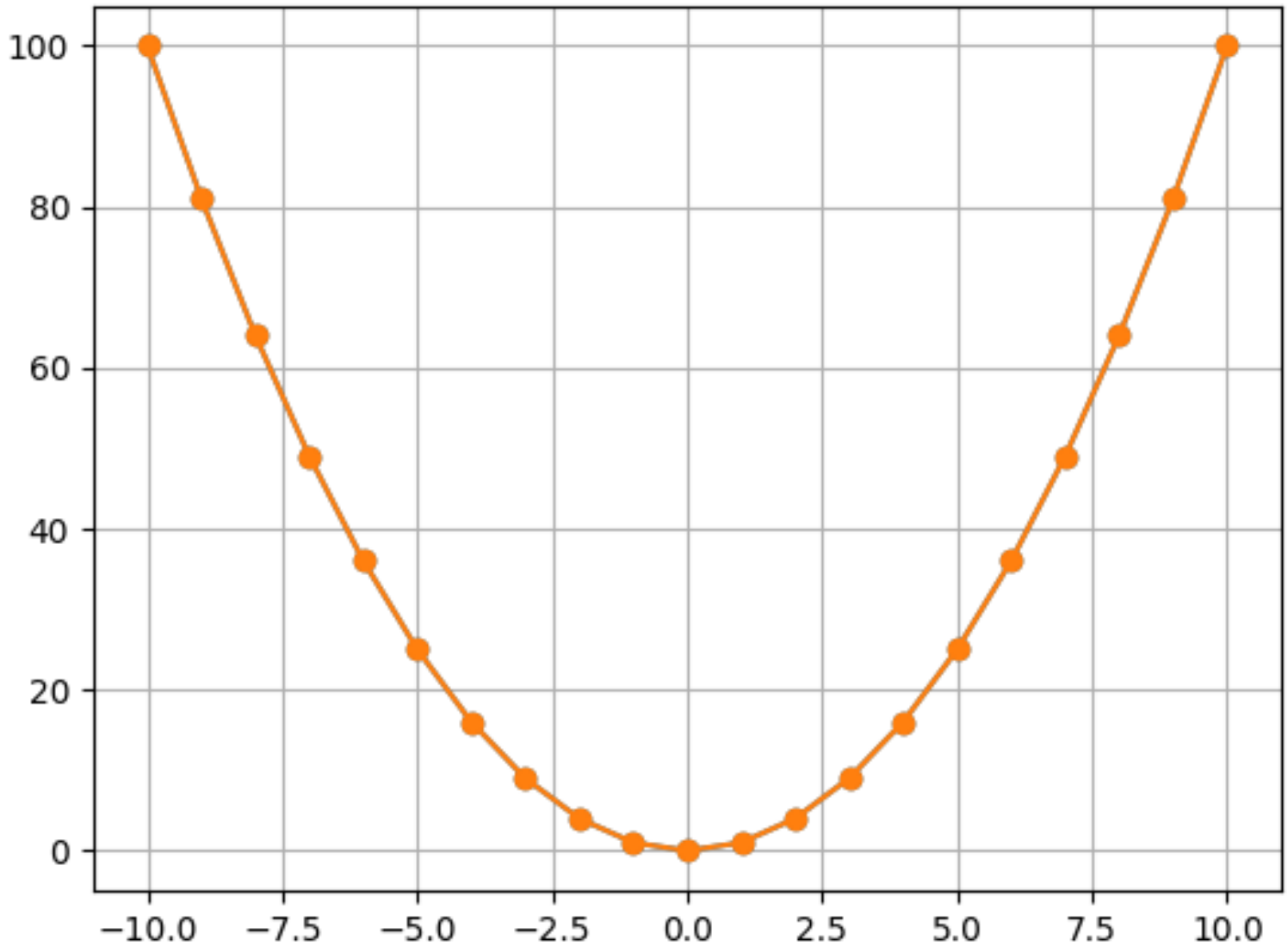
Типичный алгоритм построения графика

1. Создать графики
2. Настроить оформление графика
3. Показать или сохранить график

Пакет matplotlib

```
X = list(range(-10,11))
Y = [x**2 for x in X]
plot(X,Y,'-o')
grid(True)
show()
```

Пакет matplotlib

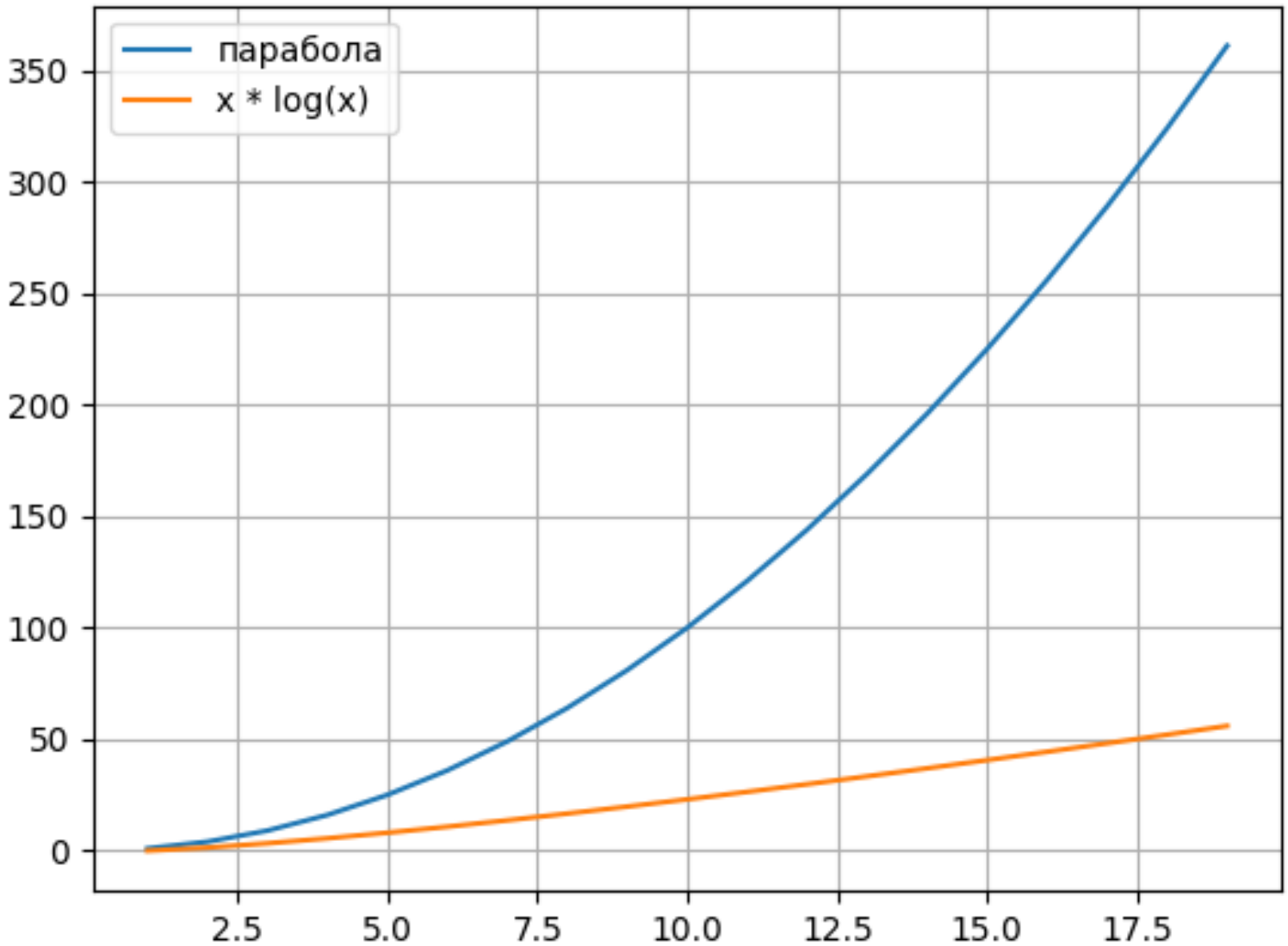


Пакет matplotlib

На одном поле можно построить несколько графиков.
Чтобы их различать стоит указать для них названия.
Для этого будем явно задавать имя параметра: **label**

```
X = list(range(20))
Y1 = [x**2 for x in X]
Y2 = [x * log(x) for x in X]
plot(X,Y, label = 'парабола')
plot(X,Y, label = 'x * log(x)')
grid(True)
show()
```

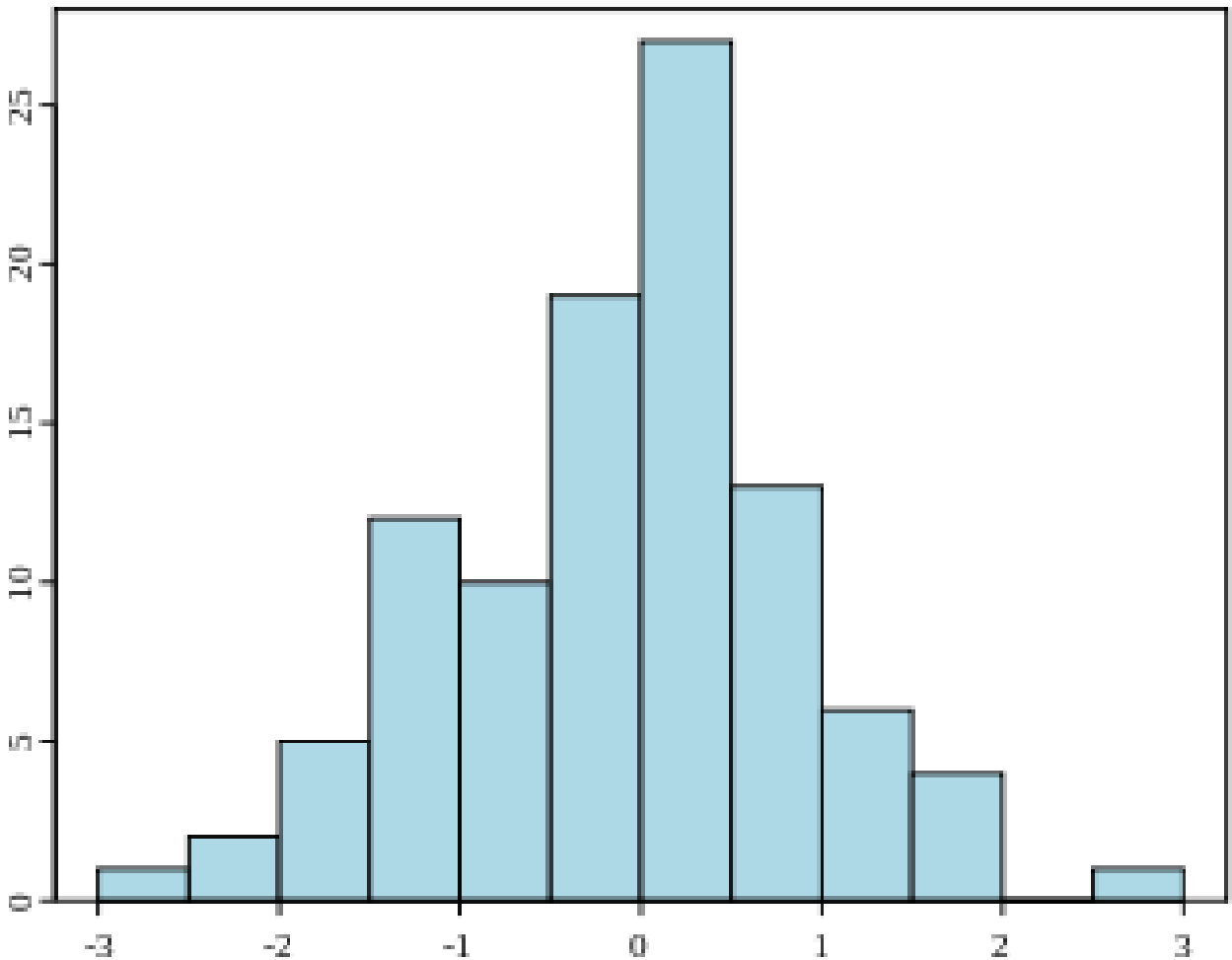
Пакет matplotlib



Пакет matplotlib

Гистограммы

Гистограмма - столбчатая диаграмма, способ графического представления табличных данных



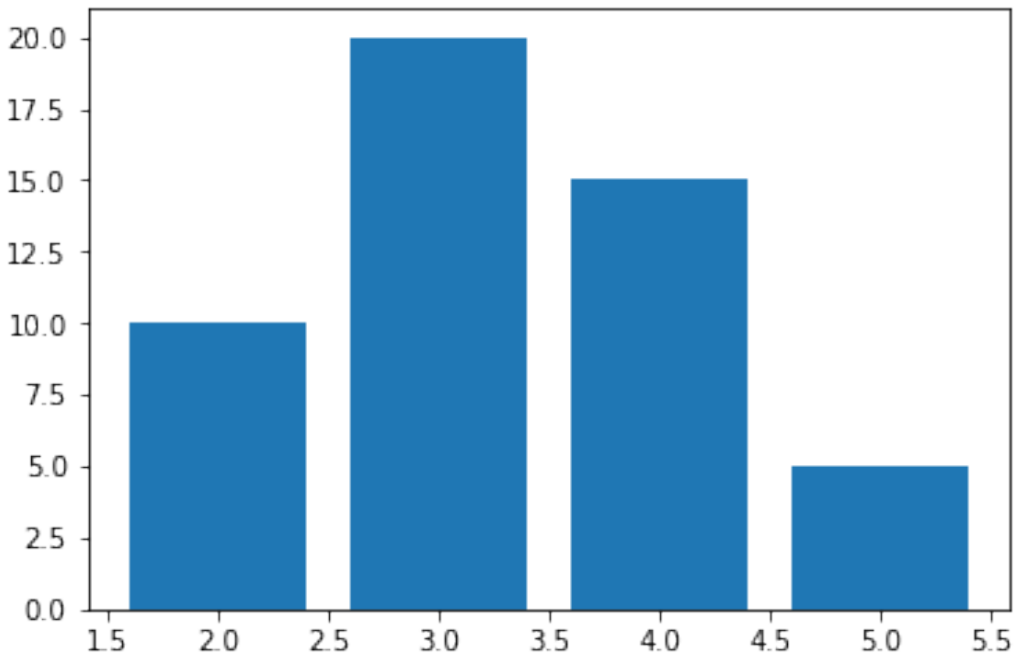
Пакет matplotlib

```
from matplotlib.pyplot import bar
```

bar(x-values, y-values) - строит гистограмму.
x-values - значения x
y-values - значения y (высоты столбцов)

Пакет matplotlib

```
X = [2,3,4,5]
Y = [10,20, 15,5]
bar(X,Y)
show()
```



Добавление дополнительных элементов на график

- Заголовок

  ```
  title('1a TITLE')
  ```

- Текст

  ```
  text(x,y, 'текст')
  ```

Пакет matplotlib

Matplotlib и Jupyter
По-умолчанию графики построенные в Jupyter будут показаны в ячейке вывода.

Чтобы показать их в отдельном окне, с возможностью масштабирования и перемещения следует перед построением графиков выполнить в Jupyter команду:

```
%matplotlib
```

Чтобы вернуть построение графиков в блокнот нужно выполнить команду

```
%matplotlib inline
```

Эстетика графиков

На основе matplotlib создано много пакетов для визуализации данных. Один из них - **sebaorn**. Этот пакет предназначен для представления статистических данных и поэтому работа с ним может показаться немного сложнее.

Графики построенные этим модулем смотрятся лучше, чем аналогичные, построенные с помощью matplotlib (со стандартными настройками)
Для того чтобы использовать стиль оформления seaborn достаточно просто подключить этот модуль и далее работать с maplotlib.

Эстетика графиков

На основе matplotlib создано много пакетов для визуализации данных. Один из них - **sebaorn**. Этот пакет предназначен для представления статистических данных и поэтому работа с ним может показаться немного сложнее.

Графики построенные этим модулем могут выглядеть лучше, чем аналогичные, построенные с помощью matplotlib (со стандартными настройками)
Для того чтобы использовать стиль оформления seaborn достаточно просто подключить этот модуль и далее работать с maplotlib.

```
import seaborn
```

Пакет matplotlib

Пакет seaborn не входит в стандартную библиотеку Python. Поэтому его придётся установить отдельно используя программу pip (поставляется вместе с Python)

```
pip.exe install seaborn
```

Outline

О параметрах функций

Формальные параметры - параметры описанные при объявлении функции.
Фактические параметры - параметры которые *фактически* передаются в функцию при её вызове. В качестве таких параметров выступают выражения.

Число и порядок формальных параметров определяют число и порядок фактических параметров.

О параметрах функций

Рассмотрим *описание* функции range приведённое в документации:

```
range(start, stop) -> range object
```

start, stop - формальные параметры.

При вызове на место формальных параметров нужно подставить фактические.

```
range( 2, 10*2 )
```

2 и 10*2 - фактические параметры.

О параметрах функций

Таким образом порядок и тип параметров, которые должны передаваться функции, чётко определён.

Функция plot не исключение. Например, если ей передано три параметра, то первый обязательно должен содержать абсциссы точек, второй - ординаты и третий - строку описывающую стиль графика.

Однако такой способ передачи параметров не всегда удобен: например нельзя оставить значение третьего параметра на усмотрение языка, передавая первый со вторым и четвёртый.

```
plot(X,Y, ,"парабола")  # SyntaxError: invalid syntax
```

Нельзя предать в функцию название графика игнорируя параметр стиля.

О параметрах функций

Каждому параметру внутри функции может быть дано отдельно имя. Тогда можно передавать параметры в функцию не в строго определённом функцией порядке, а явно указывая имя параметра.

```
function ( param-name = expr )
```

param-name - имя формального параметра.

О параметрах функций

Используя этот способ можно задавать свои значения только определённым формальным параметрам.

Например:

```
plot(X, Y, label = 'график 1')
```

```
print(".....", end="")
```

Ссылки и литература

- wikipedia: Matplotlib
- Matplotlib documentation
- Matplotlib: Научная графика в Python
- Использование библиотеки Matplotlib

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming